On the Detection of Licenses Violations in the Android Ecosystem

Ons Mlouki, Foutse Khomh, Giuliano Antoniol
SWAT–SOCCER Labs., Polytechnique Montréal, Québec, Canada
{ons.mlouki, foutse.khomh, giuliano.antoniol}@polymtl.ca

***Abstract*—Mobile applications (apps) developers often reuse code from existing libraries and frameworks in order to reduce development costs. However, these libraries and frameworks are governed by licenses to which developers must comply. A failure to comply with a license is likely to result in penalties and fines. In this paper, we analyse the licenses of 857 mobile apps from the F-droid market with the aim to understand the types of licenses that are mostly used by developers of open-source mobile apps and how these licenses evolve over time. We also investigate licenses violations and the evolution of these violations over time.**

Results show that developers of open-source mobile apps mostly use GPL and Apache licenses. We found licenses violations in 17 out of 857 apps, and 7 apps still had violations in their latest release at the time of this study. We also observed that many files are not licensed in their first release. Developers seem to have some difficulties understanding the legal constraints of licenses' terms.

***Index Terms*—Licenses violations, android applications, clone detection, libraries provenance.**

I. Introduction

When developing a new software, developers often reuse code chunks and components that have been made available under a variety of licenses (*e.g.*, Apache, BSD, GPL, or LGPL). Software licenses govern the way a software component or chunk of code can be reused or distributed. These software licenses describe the liabilities and responsibilities of parties interested in using, modifying, or redistributing software artifacts. Unlike proprietary licenses, open-source licenses allow access to the source code at any time and without any restriction. However, reuse and–or distribution are often limited by certain conditions [1]. For example, the Section 5 of the GPLv3.0 license [2] states the following about code modification: *"You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy"*. Hence, the use of components governed by different licenses into a same software system can generate a licence violation if the rules of the different licenses are not compatible. This is the case for GPLv2 and Apachev2. In fact, GPLv2 requires that any software system using a component licensed under its terms should be licensed under the GPLv2 license while the Apache Software Foundation requires that all Apache software must be distributed under the Apachev2 license. These two licenses are therefore incompatibles and any software system that contains components under these two licenses (at the same time) is exposed to penalties [3].

With the very high speed of apps development (on average 2,371 new apps are published in Google Play [4] every day), developers are increasingly inclined to reuse code from other open source projects. Vendome et al. [5] who investigated how developers of open-source projects adopt and change licenses found that these developers often experience difficulties understanding licensing terms. These factors combined with the large number of available licenses (*i.e.*, more than 70 open source licences exist today [1]) that developers could choose from, makes license violations very likely. This paper presents the results of an empirical study that aims to understand the state of license usages and violations in the Android Ecosystem. We mine the license information of 857 apps from the F-droid market and investigate license usages and evolutions. We also detect and track licenses violations overtime. Overall, we answer the following three research questions:

RQ1: What are the most common licenses used in open source mobile apps?

Among the 857 android apps analyzed in our study, when considering only the latest releases reported in F-droid, we find that 35% of apps are licensed under GPLv3, 24% under Apachev2 and 12% under the MIT License. If we take into account all the releases of each app, the numbers are a bit different, with 37% of releases licensed under GPLv3, 8% under Apachev2 and 4% under the MIT License. The difference is due to the fact that many apps change their licenses overtime. At the file level, GPL and Apache are still the most used licenses; they represent 47% and 12% of files respectively. However, there are more files licensed under the BSD license than the MIT license. We also observed that 85% of apps contain files with potential licenses violations. These potential licenses violations were mostly between GPL and non GPL licenses. We found many files licensed under a non GPL license, that contain code chunks similar to those of GPL files. We investigated the domains of the apps (*e.g.*, Games, Multimedia, office, etc...) involved in these potential violations and found that the similar files that are licensed under different terms mostly belong to apps from different domains. This suggests that developers tend to copy code from apps that are not in the same category as their app (*i.e.*, a Games app vs. a Multimedia app).

RQ2: How do mobile apps licenses evolve overtime?

In RQ1, we observed that many apps change their licenses overtime. In this research question, we analyse the evolution of licenses overtime both at project and file levels,

in order to identify the main licenses evolution patterns followed by developers of mobile apps. Our goal is to determine if overtime, developers tend to make their apps more–or–less open for reuse. Results show that developers often change the license of their mobile apps towards a less restrictive license.

RQ3: How do mobile apps licenses violations evolve overtime?

In RQ1, we observed that 85% of apps contain files with potential licenses violations. In this research question, we investigate license violations in more details and analyze their evolution overtime. Out of the 857 studied apps, we found 17 apps with clear license violations. These 17 apps have inconsistencies between their declared licenses and the licenses of their files and–or the licenses of their dependencies. We do not claim license violations for the 85% of apps that contain files with inconsistencies because although these apps contain files that share code with files licensed under a different (conflicting) license, it is possible that the two apps copied the code from a third app that released the code under a dual license, and the two apps simply picked different licenses. For the 17 projects that clearly violate license terms, 10 of them corrected the issue after 19 releases in average. The remaining 7 projects still had a license violation at the time of this study.

The remainder of this paper is organized as follows. Section II presents the motivation behind our work. Section III explains our methodology for license identification. Section IV discusses our data collection and processing. Section V describes and discusses the results of our three research questions. Section VI presents threats to the validity of our study, while Section VII summarizes related works. Section VIII concludes the paper and outlines some avenues for future works.

II. Motivating examples

The reuse of different pre-existent components to build a new system may lead to licenses violations. In March 2011, OpenLogic, a company that advocates for (and helps ensure) the proper use of open source software ran their OSS Deep Discovery license compliance scanner against 635 apps from both the Android Market and the Apple App Store, and found that 71% of the apps that were using open source code failed to comply with the terms of the open source licenses. The apps did not license their code properly. In the case of Apache licenses, information about the notice/attribution of the licenses were missing. For GPL licenses, developers of the apps failed to provide information about how to access the code. Although these developers were not pursued in court for these infringements, penalties for licensing violations can be severe. In August 2012, Samsung was found guilty of violating Apple's iPhone and iPad technologies licenses, and condemned to pay to Apple a billion dollars in damages. In march 2015, VMware was also sued by Christoph Hellwig, a Linux developer and the nonprofit organisation SFC (Software Freedom Conservancy) over an improper use of the Linux kernel [3]. Christoph and SFC claimed that VMware violated the terms of the copyright license of the Linux kernel. These few examples highlight the importance of tracking and correcting eventual licenses violations early on before the distribution of a software system.

III. Identification of Licenses Information from Mobile Apps

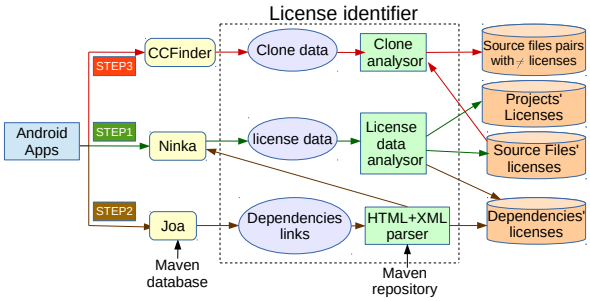

Fig. 1: Identification of license information

Our goal in this paper is to identify licenses usages and violations in the Android ecosystem. To achieve this goal, we first need to identify license information from mobile apps. License information can be found in source file as a license statement or in a separate text document generally included in the main directory of the project. Android apps also, as any project, may use some external libraries. However, those libraries are in the binary format, making it impossible to retrieve their license information through a textual analysis. In the following, we explain the steps of the approach followed in this work to retrieve the license information of mobile apps. Figure 1 summarizes the steps of this approach.

Step 1: Identification of license statements

A license statement is a textual information included in the top of each source code file or in a separate textual file often named (COPYING, LICENSE, README or POM) stored under the main repository of the project. It includes copyright information, such as the names of contributors to a source code file, the copyright owner, warranty, and liability statements. Multiple tools have been proposed in the literature to extract license's information from source code or license files (*e.g.*, README files): Ninka [6], FOSSology [7] and OSLC[1]. Table I summarizes the reported performances of these tools.

Among these three tools, we selected Ninka for our study, since it is reported to have a high accuracy and performance. To identify license information in files, Ninka splits statements from source code or license files into textual sentences, normalizes them, and matches them against known licences tokens.

[1] http://oslc.sourceforge.net/

TABLE I: Evaluation of license identification tools [6]

Tools	Recall	Precision	Execution Time
Ninka	82.3%	96.6%	22s
FOSSology	99.2%	55.0%	923s
OSLC	29.5%	29.5%	372s

TABLE II: Licensing Rules [14]

Library License	Project include	Combination must be under
GPLv2	LGPLv2.1	GPLv2 only
GPLv2	LGPLv2.1+	GPLv2 only
GPLv2+	LGPLv2.1	GPLv2+
GPLv2+	LGPLv2.1+	GPLv2+
GPLv2+	LGPLv3+	GPLv3
GPLv3	GPLv2+	GPLv3
GPLv3	LGPLv2.1	GPLv3
GPLv3	LGPLv2.1+	GPLv3
GPLv3	LGPLv3+	GPLv3
LGPLv3	GPLv2+	GPLv3

Step 2: Identification of the licenses of libraries used by Apps

Mobile apps frequently use externally developed libraries to provide their services. However, these libraries are often deployed in the form of binaries, making it impossible for Ninka, or any of the above mentioned tools to extract licenses information from these libraries. To address this issue, we make use of the JOA tool proposed by Davies et al. [8]. JOA allows to track the provenance of software artifacts (both source code and binaries). In fact, Davies et al [8] downloaded libraries from the Maven repository, computed signatures for these libraries and stored them in a database. They also developed the tool JOA that can compute the signature of any new library and match it against the database to identify the identity of the new library. Knowing the identity of the libraries used by our studied mobile apps allows us to extract their license information by parsing the POM files available in the Maven repository. In this study we use the same database as Davies et al [8], which contains all libraries that were available in the Maven repository in April 2013.

Step 3: Detection of similarities between the files in mobile apps

Even though Ninka does a great job identifying licenses from licenses statements, we should not forget that license statements are just blocks of comments in some files. Hence, they are subject to modifications. To compensate for potential inaccuracies in the licenses information extracted from source and license files using Ninka, we use clone detection techniques to identify source files pairs that share similar code, but are not under the same license.
Among multiple clone detection tools that exist, we choose to work with CCFinderX [9] since it is known to scale well. CCFinderX represents the content of source code files as sequences of tokens and uses a suffix-tree matching algorithm to compute matchings. Clone location information is represented as a tree with sharing nodes for leading identical subsequence. It gives as a result a list of clone pairs.

IV. Study Design

This section presents the data collection and data processing of our case study, which aims to address the following three research questions:

- RQ1: What are the most common licenses used in open source mobile apps?
- RQ2: How do mobile apps licenses evolve overtime?
- RQ3: How do mobile apps licenses violations evolve overtime?

A. Data Collection

We crawled the F-droid Website and downloaded 857 mobile apps from the android market F-Droid. We chose these apps because their source code is available on Github[2]. Next, for each app, we downloaded all its releases from the Github repository. Figure 2 shows the distribution of our data across the categories of apps.

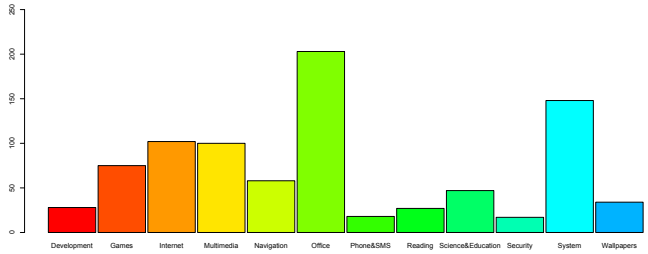

Fig. 2: F-Droid categrories

B. Data Processing

Figure 3 shows an overview of our data processing approach. First, we used our license identification approach described in Section III to extract licenses information for all the 857 apps. Next, we built four databases containing respectively, Android apps licenses, source files licenses, dependencies (*i.e.*, libraries) licenses and pairs of cloned files found with different licenses. To track licenses violations, we implemented a script that checks for licenses inconsistencies using information from Table II and Table III . Then, we ran this script against our four databases to obtain information about licenses violations.

V. Case Study Results

This section presents and discusses the results of our three research questions. For each question, we present the motivation, the approach followed to answer the question, and the findings.

[2] http://github.com

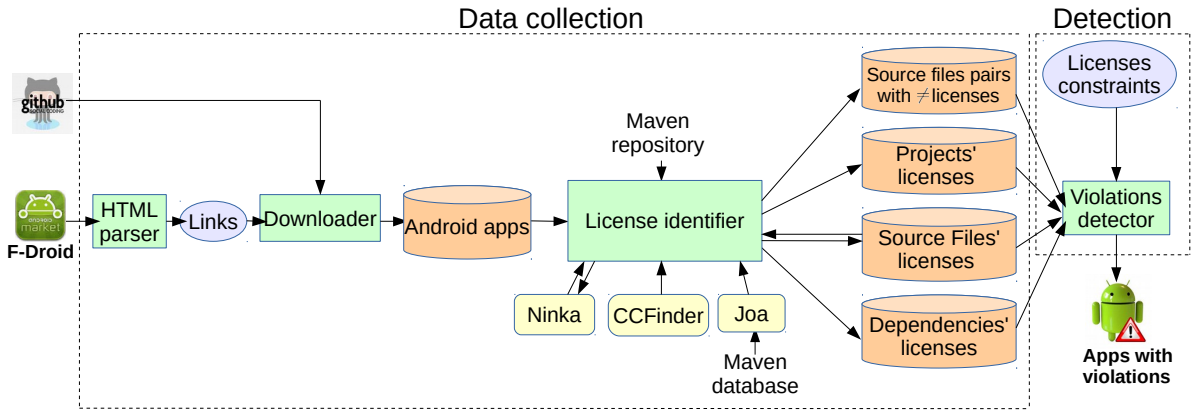

Fig. 3: Overview of our approach to identify applications with license violation in an android market

RQ1: What are the most common licenses used in open source mobile apps?

Motivation.

This research question is preliminary to the others and aims to identify licenses that are frequently used by developers of Android apps. The results of this research question will provide insights about the preferences of mobile apps developers among the more than 70 open source licenses that are available.

Approach.

To answer this research question, we perform our analysis both at file and project levels following the approach described in Section III. We extract the licenses information of a project and its source files using the Ninka tool. We use the JOA tool to identify the provenance of libraries used by the android apps; tracing them to the Maven repository. Then, by analysing their POM files, we obtain their license information. To identify licenses inconsistencies at file level, we use the CCFinderX clone detection tool.

Findings.

F-Droid provides licence information only for the latest release of each app. When considering only this information, **we obtain that 35% of apps are licensed under GPLv3, 24% under Apachev2 and 12% under MIT License.** Figure 4 presents the licenses distribution of the final releases of all the apps in our data sets.

When considering all the released versions of each app from our data set, the picture is a bit different. **We obtain that 37% of releases are licensed under GPLv3, 8% under Apachev2 and 4% under MIT License.** This difference is due to the fact that many apps change their licenses overtime. Figure 5 presents the distribution of licenses for all the releases of all the apps in our data set. We can observe that more than 3,250 apps releases out of 8,938 apps releases are unlicensed or their license information is not declared in any of our analysed files (described in Section III).

At the file level, GPL and Apache are still the most used license; representing 47% and 12% of files respectively.

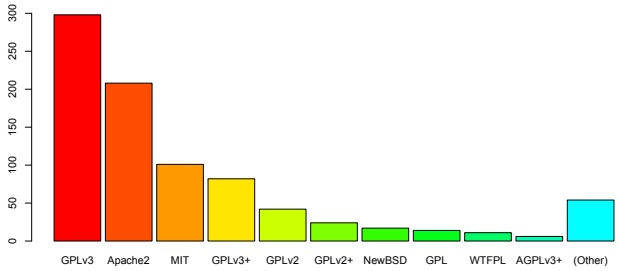

Fig. 4: Projects licenses when considering only the latest release of each app

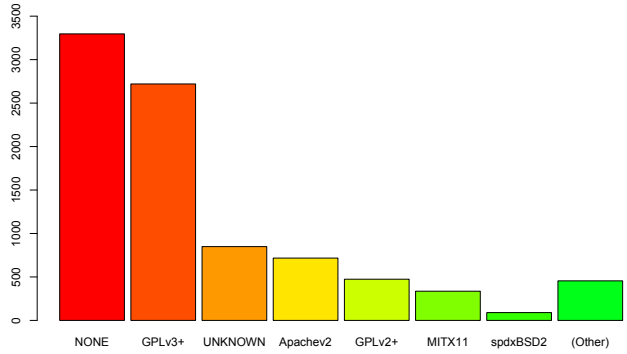

Fig. 5: Projects licenses when considering all the releases of the apps

However, there are more files licensed under the BSD license than the MIT license. Table IV summarizes results obtained at the file level. Our set of apps contained in total 1,429,330 source code files (1,168,899 written in Java; 81,378 written in C; and 138,396 C++ files). In Table IV the keywords

TABLE III: License incompatibilities

Incompatible license	Reason for the incompatibility
Apachev2 vs. GPLv2	Copyleft licenses are “reciprocal”, “share-alike”, or “viral” license each of them says, “If you include code under this license in a larger program, the larger program must be under this license too.”[10]
Apachev2 vs. GPL (except GPLv3)	restrictions of copyleft licenses; each of them says, “If you include code under this license ... the larger program must be under this license too.” “Apachev2 software can ... be included in GPLv3 projects”
Apachev2 vs. GPLv3	Copyleft licenses But one way is permitted; project under GPL3 source may have Apachev2[10]
Apachev2 vs. LGPL 2, 2.1, 3	“The LGPL is ineligible primarily due to the restrictions it places on larger works, violating the third license criterion. Therefore, LGPL-licensed works must not be included in Apache products”[11]
GPLv2 vs. GPLv3+	restrictions of copyleft licenses; each of them says, “If you include code under this license ... the larger program must be under this license too.”
GPLv2 vs. GPLv3	restrictions of copyleft licenses; each of them says, “If you include code under this license ... the larger program must be under this license too.”
LGPLv3+ vs. GPLv2	[12]
GPL vs Apachev1.0	“a lax, permissive non-copyleft free software license with an advertising clause. This creates practical problems like those of the original BSD license, including incompatibility with the GNU GPL” [12]
GPL vs Apachev1.1	“a permissive non-copyleft free software license. It has a few requirements that render it incompatible with the GNU GPL, such as strong prohibitions on the use of Apache-related names” [12]
GPL vs CDDL	http://www.gnu.org/licenses/license-list.en.html “It has a weak per-file copyleft. This means a module covered by the GPL and a module covered by the CDDL can not legally be linked together.” [13]
GPL vs CPLv1	“its weak copyleft and choice of law clause make it incompatible with the GPL” [12]
GPL vs EPLv1	“its weak copyleft and choice of law clause make it incompatible with the GPL” [12]
GPL vs MPLv1.1	“a free software license which is not a strong copyleft; unlike the X11 license, it has some complex restrictions that make it incompatible with the GNU GPL” [12]
GPL vs BSD4	“a lax, permissive non-copyleft free software license with a serious flaw: the “obnoxious BSD advertising clause”.” [12]

“SeeFile”, “UNKNOWN” and “ERROR” are used by Ninka to indicate that the analyzed file points to another file that may contain the license, that the found license is not recognized, and that Ninka encountered an error.

We tracked the licenses of libraries used by our studied apps and obtained the distribution shown in Figure 6. There we report only libraries that perfectly matched a known library from the Maven repository; *i.e.*, the proportion of files in the library that matched the files contained in a library from the Maven repository is 100%. We opted for this conservative approach because if a library used by an app is licensed under Apachev2 and only matches 50% of the files in a library from the Maven repository that is licensed under GPLv3, we cannot know for sure if the app is using the part of the features provided by the files under the GPLv3 license, and hence, we cannot conclude on a potential license violation. In fact, a GPLv3 library can contain files licensed under Apachev2 (since a project containing Apachev2 sources code and GPLv3 source code must be licensed under GPLv3) and the app may be using only the Apachev2 files.

We applied clone detection to identify potential inconsistencies between the license statements reported in files. License inconsistencies are a symptom of licenses violations, since each license is protected by a different copyright. Table V summarizes inconsistencies found among the licenses statements of our studied apps. We investigated the domains of apps involved in these inconsistencies and found that the similar files that are licensed under different terms mostly belong to apps from different domains (as presented in Table VI). This result suggests that developers tend to copy code from apps that are not in the same category as their app. This was the case for 60,72% of the inconsistencies found.

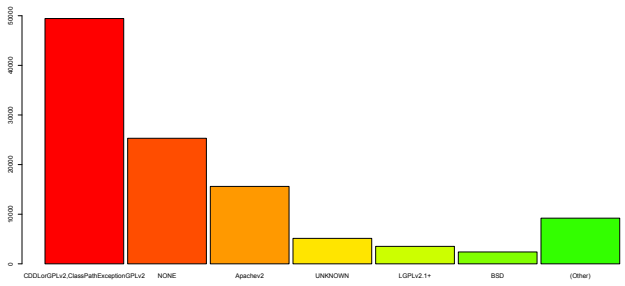

Fig. 6: Licenses of libraries used by the studied apps

TABLE IV: Licenses detected at file level

License	Version	# Occurrence	%
GPL	noVersion	32	**45,68**
	v2	37 388	
	v2+	395 198	
	v3	19 125	
	v3+	201 171	
Apache	Apache	2	**12,39**
	v1.0	52	
	v1.1	6	
	v2	184 765	
LGPL	v2	84	1,13
	v2.1	2 962	
	v2.1+	7 634	
	v2+	95	
	v3	2 097	
	v3+	3 284	
BSD	3NoWaranty	2 279	1,07
	BSD3	6 759	
	BSD4	168	
	spdxBSD2	2 281	
	spdxBSD3	3 655	
	spdxBSD4	130	
MIT	oldwithoutSelland	5	0,87
	oldwithoutSelland NoDocumentationRequi	147	
	MITVariant	1	
	MITX11	11 965	
	X11BSDvar	4	
	X11noNotice	343	
PublicDomain		2 957	0,21
artifex		136	0,01
BeerWareVer42		6	0
CDDLorGPLv2		475	0,03
CPLv1		20	0
DoWTFYWv2		48	0
EPLv1		13	0
FreeType		1 714	0,12
MPLv1_1		4	0
ZLIBref		137	0,01
SunSimpleLic		555	0,04
SimpleLicence1		18	0
orGPLv2+orLGPLv2.1+		234	0,02
None		392 314	**27,45**
SeeFile		8 053	0,56
UNKNOWN		139 484	9,76
ERROR		930	0,07
Total number of source code files analysed		**1 429 330**	

RQ2: How do mobile apps licenses evolve overtime?

Motivation.

Our first research question showed that developers often change the license of their mobile apps after a few releases.

TABLE V: Inconsistencies found among the license statements of the studied apps

Kind of disagreement	#Files	#Releases	#Apps
GPLv2-Apachev2	54 335	1 067	70
GPL-OTHER	18 853 940	7 141	685
Others	18 284 150	-	-
Apps with license inconsistency	-	-	731

TABLE VI: License inconsistencies by categories

Category	Part of desagreement (%)
Inconsistent files from different domains	60,72
SystemApps	1,09
OfficeApps	14,69
InternetApps	16,62
ScienceAndEducationApps	3,65
ReadingApps	1,75
GamesApps	0,08
MultimediaApps	0,29
DevelopmentApps	0,09
NavigationApps	0,73
PhoneAndSMSApps	0,28
SecurityApps	0,01

In this research question, we analyse the evolution of licenses overtime both at project and file levels, in order to understand the main patterns of licenses evolution in the Android Ecosystem.

Approach.

For each file in each app, we track the evolution history of the file and build a genealogy. We identify files across the releases using their absolute paths. To handled cases of renaming, we apply clone detection to track files with similar contents that were renamed. Next, using licenses information collected in RQ1, we map licenses to the different versions of each file and compute all licenses evolution patterns. Finally, we build state transition models capturing the license evolution patterns at file and project levels, respectively. For each transition in our state-transition diagrams, we compute the probability of the transition following Equation 1. To focus our interest, we narrow the data analyzed to only entities that experienced at least one change in their life-cycle. Thus, to calculate the probability of a transition from License A to License B, we calculate the occurrence of License A $\rightarrow$ License B divided by the occurrence of License A in our reduced data set.

$$\mathbf{OccurrenceOf(A \rightarrow B)} \,/\, \mathbf{OccurrenceOf(A)} \quad (1)$$

Findings.

1) License changes at the file level: Among the 857 apps from our data set, 128 apps experienced a license change.

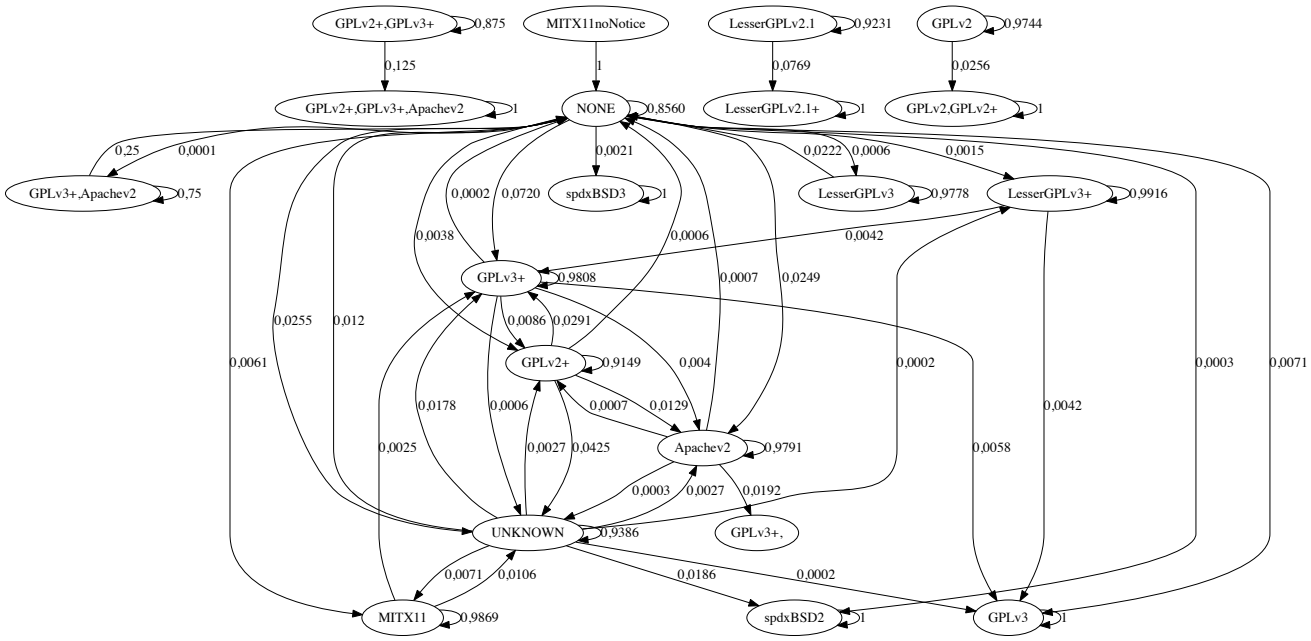

Fig. 7: Evolution of licenses at the file level

These 128 apps contain 2,062 Java files that experienced at least a license change during their app's life-cycle. Figure 7 summarizes the transition patterns found in our data set and their probabilities. As one can see, the probability for a file to stay under the same license is very high for almost all the licenses (0.9 in average). We classify our patterns into two categories: general patterns (summarized in Table VII), which contains patterns that include no license or unknown license; and specific patterns (summarized in Table VIII), which contains only patterns where we have changes between two known licenses. In the following, we discuss each of these categories in more details.

General patterns: files in this category generally start with no License (NONE) in their first release. This finding is consistent with observations made by Vendome et al. [5] that developers tend to delay their decision about license selection. Regarding transitions from a known license to NONE or UNKNOWN license, we could not find a plausible explanation, after manually examining all these patterns. It is unclear why some developers remove the license of their app. Below are

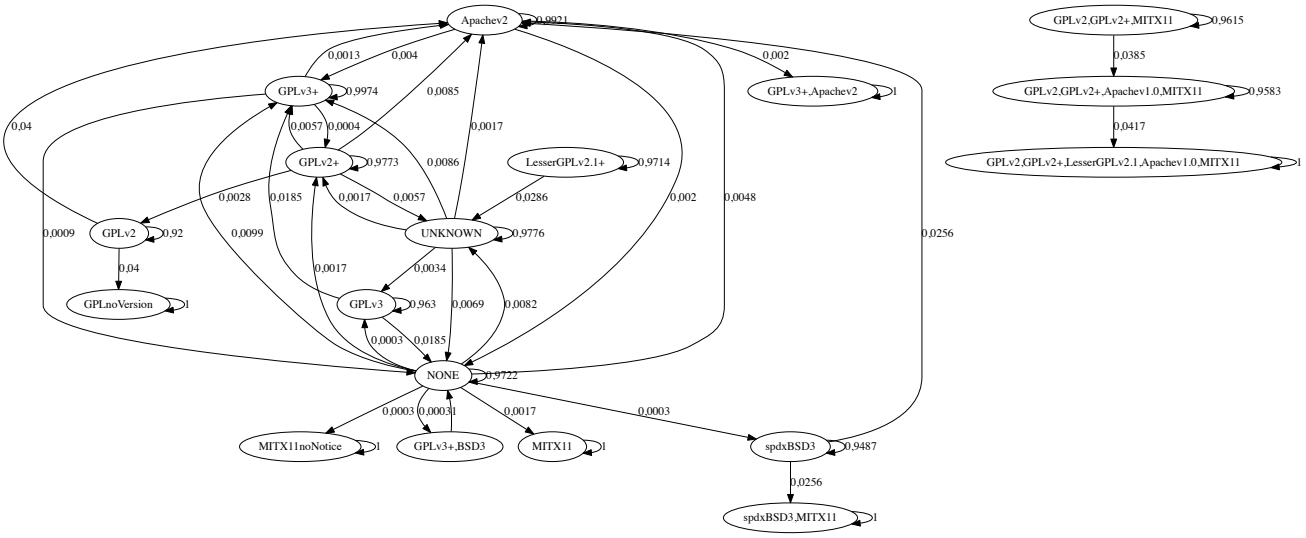

Fig. 8: Evolution of licenses at the projects level

TABLE VII: General patterns

Pattern	#Occurence
UNKNOWNorNONE → GPL3+	820
UNKNOWNorNONE → Apachev2	269
UNKNOWNorNONE → MITX11	92
UNKNOWNorNONE →spdxBSD2	79
GPL2+ → UNKNOWNorNONE	77
UNKNOWNorNONE → GPL3	75
UNKNOWNorNONE → GPL2+	50
NONE → spdxBSD3	22
UNKNOWNorNONE → LGPLv3+	17
MITX11 → UNKNOWN	17
GPL3+ → UNKNOWNorNONE	10
NONE → LGPLv3	6
Apachev2 → UNKNOWNorNONE	3
LGPLv3 → NONE	1
MITX11noNotice → NONE	1
GPLv3+,Apachev2 → NONE	1
NONE → GPLv3+,Apachev2	1

TABLE VIII: Specific patterns

pattern	#Occurrence
GPLv3+→GPLv2+	109
GPLv3+→GPLv3	73
Apachev2→GPLv3+	56
GPLv3+→Apachev2	50
GPLv2+→GPLv3+	52
GPLv2+→Apachev2	23
GPLv2→GPLv2,GPLv2+	5
MITX11→GPLv3+	4
Apachev2→GPLv2+	2
LGPLv2.1→LGPLv2.1+	2
LGPLv3+→GPLv3+	1
LGPLv3+→GPLv3	1
GPLv2+,GPLv3+→GPLv2+,GPLv3+,Apachev2	1

some examples of general patterns found: The app *PageTurner* started with no license. Later on developers licensed it to GPLv3+ and changed all its Apachev2 files to GPLv3+.
Wifi-Fixer also didn't have any license initially but later on was licensed under GPLv3+ and all its Apachev2 files were licensed under GPLv3+.
AndroidCaldavSyncAdapater was initially under no license, developers later on changed all its Apachev2 files to GPLv3+ and licensed the app under GPLv3+. All other source files in that app are under BSD3.
Specific patterns: We examined transitions between known licenses and noticed cases of license upgrade and downgrade. We discuss some of these cases in the coming paragraphs. The app *keepassdroid* was initially released under the GPLv2 license. However it contained files licensed under Apachev2 and GPLv2 licenses which are incompatible. Later on, the project was changed to the dual license "GPLv2, GPLv2+" in order to solve its license violation.
The app *open-keychain* was initially licensed under Apachev2, then, developers changed all Apachev2 Java files to the GPLv3+ license and updated the license of the app to GPLv3+.
The app *geopaparazzi* is now under GPLv3+. It started with no license, then, developers changed some Apachev2 files to the GPLv3+ license. The app also contain files under LesserGPLv2.1+ and GPLv3 licenses. The last 15 releases of the app (out of a total of 79 releases) are under GPLv3+.
The *connectbot* app started under the GPLv3+ license. It maintained this license through 11 releases. It contains files under BSD3, BSD, GPL2+, MITX11, GPLv3+ and apachev2 licenses. The license of *connectbot* was changed to Apachev2 in the last 22 releases (out of a total of 33 releases). During this transition, developers changed the license of all GPLv3+ files to Apachev2.
gnucash-android was first released under GPLv2+, files were under GPLv2+ and Apachev2 licenses, then, the license of the app was changed to Apachev2 and all GPLv2+ files were licensed under Apachev2.

2) Licenses changes at the project level: We noticed that the probability for a project to stay under the same license is very high (see Figure 8). Only 122 apps out of 857 apps changed their license at least once during their lifetime. Apps license changes are generally toward more permissive licenses (see Table IX)

RQ3: How do mobile apps licenses violations evolve overtime?

Motivation. In RQ1, we found 85% of apps containing files with licenses inconsistencies. However, not all of these inconsistencies are likely to be license violations. In fact, since it is possible to release code under a dual license (*e.g.*, Apachev2 and GPLv2), two apps reusing a code licensed under a dual license could simply chose to pick different licenses (*e.g.*, the first app picking Apachev2 and the second app GPLv2). Our clone analysis described above would report this as a case of license inconsistency. However, there is no license violation in this case. In this research question, we aim to identify clear cases of licenses violations in the Android free software apps ecosystem and examine if and–how developers address these violations.
Approach.
To track license violations, we adopt a conservative approach. We consider that an app violates some license terms only when there are inconsistencies between the declared license of the app and the licenses of its files, among its files and–or the license of its dependencies.
Findings. Table X shows the proportion of license violations found in our data set. Among the 857 studied apps, 17 apps (totaling 229 releases) clearly violate the terms of some open-source licenses. From the 17 apps that contain a license violation, only 10 apps solved the license violations after some releases. The remaining 7 projects still had a license violation at the time of this study. To solve license violations, developers either changed the licenses of some of the app's files or removed the contentious files from the apps. For the 10 apps that solved licenses violations. We noticed that it took them on

TABLE IX: License change patterns at project level

Patern	Occurence
General Patern	
UNKNOWNorNONE→GPLv3+	40
UNKNOWNorNONE→Apachev2	16
GPLv3+→UNKNOWNorNONE 7	7
UNKNOWNorNONE→GPLv2+	6
NONE→MITX11	6
UNKNOWNorNONE→GPLv3	4
GPLv2+→UNKNOWN	2
UNKNOWNorNONE→spdxBSD2	2
NONE→MITX11noNotice	2
NONE→spdxBSD3	2
NONE→GPLv3+,BSD3	1
Apachev2→NONE	1
GPLv3→NONE	1
GPLv3+,BSD3→NONE	1
LesserGPLv2.1→UNKNOWN	1
Specific Pattern	
GPLv3+→Apachev2	3
GPLv2+→Apachev2	3
Apachev2→GPLv3+	2
GPLv2+→GPLv3+	2
Apachev2→GPLv3+,Apachev2	1
GPLv3→GPLv3+	1
GPLv3+→GPLv2+	1
GPLv2+→GPLv2	1
GPLv2→Apachev2	1
spdxBSD3→spdxBSD3,MITX11	1
spdxBSD3→Apachev2	1
GPLv2→GPLnoVersion	1
GPLv2,GPLv2+,MITX11→GPLv2,GPLv2+,Apachev1.0,MITX11	1
GPLv2,GPLv2+,Apachev1.0,MITX11 →GPLv2,GPLv2+,LesserGPLv2.1,Apachev1.0,MITX11	1

average 19 releases to correct the violation. This means that 19 releases of some apps were distributed into the Android market with license violations, which is quite troubling.

VI. Threats to Validity

This section discusses the threats to validity of our study following the guidelines for case study research [15].

Construct validity threats concern the relation between theory and observation. In this study, the construct validity threats are mainly due to measurement errors. We rely on Ninka, a state-of-the-art tool for license identifications in textual files. It has 97% precision and 82% recall. For our license inconsistency detection, we rely on CCFinderX for clone detection. Although, it doesn't have a 100% precision and recall, CCFinderX has the advantage that it can process a large amount of data in a reasonable time [16]. It has been used successfully in previous studies on clone detection [17]. The precision of the JOA tool has a limited impact on our results since we considered only libraries that were perfectly matched. To the best of our knowledge, JOA is the only tool that can detect the provenance of software artifacts. Our study relies on a copy of the Maven repository that was obtained in 2013, therefore, it is possible that some libraries are missing. However, it is no more possible to obtain a full copy of the Maven repository. Nevertheless, it is possible that the license of some of the libraries from our Maven repository copy were changed after 2013.

Internal validity do not affect this particular study, being an exploratory study [15]. Thus, we cannot claim causation, but just relate the occurrence of license violations, although our discussion tries to explain why license violations occurred. Also, to track file's license changes, we relied only on the absolute path of the file. However, we required that the files in the subsequent releases share a significantly large similarity (measured using CCFinderX).

Conclusion validity threats concern the relation between the treatment and the outcome. License terms could be interpreted in deferent ways. There are ongoing discussions among developers about the interpretations of the legal terms of some open-source licenses. In our study, we relied only on the information explicitly written in the official websites of the licenses. This conservative approach may limit us from detecting more incompatible licenses. However, our proposed approach to detect license violations can be easily extended to include more license rules.

External validity threats concern the possibility to generalise our results. Although we only conduct our case study with 857 mobile apps from the F-droid market, most of our findings are consistent with previous studies (*e.g.*, [5]). We share our data and scripts at: http://swat.polymtl.ca/data/SANER16/AndroidAppsDataONF-DroidJanv2015.7z. Further studies with different sets of mobile apps (including close source apps) from different markets are required to verify our results and make our findings more general.

VII. Related Work

In this section, we introduce some related literature about licenses inconsistency issues.

In [1], the authors manually examined 124 OSS packages to understand the way in which developers solve license incompatibilities. They built a model to document integration patterns that are used to help developers to solve license inconsistency issues and highlighted the characteristics of certain licenses and their applicability.

[18] proposes a method to detect licenses incompatibilities in software packages. They compared the declared licenses of binary packages with the licenses of their source files and dependencies, to identify possible inconsistencies. They validated their approach on 3,874 binary packages from the Fedora-12 GNU/Linux distribution and identified the presence of license inconsistencies. Companies, like Black Duck[3] and HewlettParckard[4] proposes their own infrastructure to help

[3] http://blackducksoftware.com
[4] http://www.hp.com

TABLE X: Projects with violation

Category	Name	License	Nb releases	releases with Violation	Kind of violation
Security	afwall	UNKNOWN	31	5 - 19	GPLv2vsApachev2 GPLv2vsGPLv3+
PhoneAndSMS	batphone	GPLv3+	37	16 - 23	GPLv2vsApachev2 GPLv2vsGPLv3+
Navigation	MozStumbler	UNKNOWN	86	58 - 86	Apachev2vsGPLnoVersion
Development	Vimtouch	Apachev2	22	20 - 22	(proj)Apachev2vsGPLv3+(source)
	Travis-Jr	Apachev2	2	1 - 2	(proj) Apachev2vsLGPL (lib)
	servdroid	Apachev2	5	1 - 5	(proj) Apachev2vsLGPL (lib)
	andlytics	Apachev2	26	24 - 26	(proj) Apachev2vsLGPL (lib)
	BalandruinoAndroidApp	GPLv2	10	1 - 4	(proj) GPLv2vsApachev2(source)
Games	ppsspp	UNKNOWN	24	1 - 13	Apachev1.0vsGPLv3+ Apachev1.0vsGPLv2+
	dolphin	NONE	9	1,3,5 - 7 1 - 7	Apachev1.0vsGPLv3+ Apachev1.0vsGPLv2+
Internet	open-keychain	Apachev2	30	1 - 4	(proj)Apachev2vsGPLv3+(source)
	frostwire-android	GPLv3+	64	1 - 32 1 - 24 13	GPLv2vsApachev2 GPLv2vsGPLv3+ (source)GPLv2vsApachev2(lib) GPLv2vsLGPLv3+
	android	GPLv2	38	15 - 31	(proj)GPLv2vsApachev2(source)
Office	Keepassdroid	GPLv2, GPLv2+, LGPLv2.1, Apachev1.0, MITX11	144	1 - 90 36 - 90	GPLv2vsGPLv3+ GPLv2vsApachev2
System	host-editor-android	Apachev2	6	2	(proj) Apachev2vsLGPL (lib)
	gogodroid	GPLv2	7	1 - 7	Apachev1.1vsGPLv2+ (proj) GPLv2vsApache
	920-Text-Editor	GPLv3+	2	1 - 2	GPLv2vsGPLv3+ GPLv2vsApachev2

users avoid license incompatibility issues. They build their own databases and validate that it does not contain any license violations. The databases are then provided to users who can reuse code from them without any fear of license violation. Although this technique solves license incompatibility issues and makes developers feel safe, it is quite limited by the scope of the databases that are built. In [19], authors introduced BAT, a tool that detects code cloning in binaries. They implemented three binary clone detection techniques. The tool helps users to detect clones between a subject binary and the binaries from their repository. Although in theory BAT could track license violations, in practice, its accuracy is poor.

VIII. Conclusion

This paper reports the results of a large empirical study aimed at understanding licenses usages and violations in the Android Ecosystem. The results of the study suggests that developers of mobile apps frequently fail to comply with the terms of licenses. Even more problematic is the fact that licenses violations persist through multiple releases of the apps before they are eventually resolved. Developers seem to lack proper knowledge about licenses management or it may be that the lack of consistency and standardization in license declarations fosters confusion among developers (as suggested by Vendome et al. [5]). Indeed, the fact that some licenses are contained in source code heading comments, while others are put in separate license files or README documents may cause developers to miss some license information. To help address this issue, we advocate for the development of tools that can assist developers in the management of licenses throughout the lifecycle of their apps. In the future, we plan to investigate further our fining that app developers tend to copy the code of apps from different domains, in order to understand the root causes of this behaviour.

References

[1] D. M. German and A. E. Hassan, "License integration patterns: Addressing license mismatches in component-based development," in *ICSE -09: Proceedings of the 31st International Conference on Software Engineering*. IEEE, 2009, pp. 188–198.

[2] "GNU General Public License," http://www.gnu.org/licenses/gpl.html, 2015, online; accessed October 05th, 2015.

[3] "lawsuit," http://www.infoworld.com/article/2893695/open-source-software/vmware-heading-to-court-over-gpl-violations.html, 2015, online; accessed April 15th, 2015.

[4] J. Koetsier. (2013, August) 700k of the 1.2m apps available for iphone, android, and windows are zombies: Last accessed: November 14, 2015. [Online]. Available: http://venturebeat.com/2013/08/26/700k-of-the-1-2m-apps-available-for-iphone-android-and-windows-are-zombies/

[5] C. Vendome, M. Linares-Vásquez, G. Bavota, M. Di Penta, D. German, and D. Poshyvanyk, "When and why developers adopt and change software licenses," in *Proceedings of the 31st IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2015.

[6] D. M. German, Y. Menabe, and K. Inoue, "A sentence-matching method for automatic license identification of source code files," in *Proceedings of the IEEE/ACM international conference on Automated software engineering*. ACM, 2010, pp. 437–446.

[7] R. Gobeille, "The fossology project," in *Proceedings of the 2008 international working conference on Mining software repositories*. ACM, 2008, pp. 47–50.

[8] J. Davies, D. M. German, M. W. G. Godfrey, and A. Hindle, "Software bertillonage: Finding the provenance of software development artifacts," *Empirical Software Engineering*, vol. 18, no. 6, pp. 1195–1237, 2013.

[9] T. Kaniya, S. Kusumoto, and K. Inoue, "Ccfinder: a multilinguistic token-based code clone detection system for large scale source code," *Software Engineering, IEEE Transactions on software engineering*, vol. 28, no. 7, pp. 654–670, 2002.

[10] "Apache License vs GPLv3," http://www.apache.org/licenses/GPL-compatibility.html, 2015, online; accessed September 17th, 2015.

[11] "Apache License Legal," http://www.apache.org/legal/resolved.html, 2015, online; accessed September 17th, 2015.

[12] "GNU General Public License Legal," http://www.gnu.org/licenses/license-list.en.html, 2015, online; accessed September 09th, 2015.

[13] "CDDL License," http://www.whitesourcesoftware.com/top-10-cddl-license-questions-answered/, 2015, online; accessed September 29th, 2015.

[14] "GNU General Public License Legal," http://www.gnu.org/licenses/gpl-faq.html#AllCompatibility, 2015, online; accessed September 09th, 2015.

[15] R. K. Yin, *Case Study Research: Design and Methods - Third Edition*, 3rd ed. SAGE Publications, 2002.

[16] S. Dang and S. A. Wani, "Performance evaluation of clone detection tools," *INTERNATIONAL JOURNAL OF SCIENCE AND RESEARCH (IJSR)*, pp. 1903–1906, 2015.

[17] L. Barbour, F. Khomh, and Y. Zou, "Late propagation in software clones," in *Software Maintenance (ICSM), 2011 27th IEEE International Conference on*, Sept 2011, pp. 273–282.

[18] D. M. German, M. Di Penta, and J. Davies, "Understanding and auditing the licensing of open source software distributions," in *Program Comprehension (ICPC), 2010 IEEE 18th International Conference on*. IEEE, 2010, pp. 84–93.

[19] A. Hemel, K. T. Kalleberg, R. Vermaas, and E. Dolstra, "Finding software license violations through binary code clone detection," in *Proceedings of the 8th Working Conference on Mining Software Repositories*. ACM, 2011, pp. 63–72.